NOTES ON THE RSA CRYPTO SYSTEM

ERIC MARTIN

1. Principle

I want to get personal data from you (e.g., your credit card number). You agree to send me the data but only if you can encrypt it and are confident that nobody but me will know how to decrypt it. Assume that the data I want from you is a natural number m (m for *message*).

- I send you a *public key* in the form of two natural numbers N and e (e for *encrypt*) and I ask you to encode your message as $m^e \bmod N$ and send me that number, say $\widetilde{m}$.
- I let you know that I have a *private key* in the form of two natural numbers, the same number N and a second number d (d for *decrypt*), I will compute $\widetilde{m}^d \bmod N$, and it will turn out that this will be precisely equal to m.

I can send you (N, e) by email, you do not have to be careful and keep it to yourself, anyone can know the public key, so anyone can encode messages as you will. But you have to be confident that the private key is known only to me and I will keep it in a safe place. Also, you have to be confident that it is just too hard to compute the private key from the public key.

What conditions can we impose on N, e and d for the scheme to work?

- Take for N the product of two prime numbers p and q. Also, N should be greater than m. Set $\phi(N) = (p-1)(q-1)$ (ϕ is known as *Euler's totient function*).
- Take for e a number smaller than $\phi(N)$ which is relatively prime to $\phi(N)$; so this condition is satisfied by any prime number smaller than $\phi(N)$.
- Take for d the (unique) number smaller than $\phi(N)$ such that $ed = 1 \bmod \phi(N)$ (the *multiplicative inverse* of e modulo $\phi(N)$).

It is easy to compute d from e and $\phi(N)$, hence for the scheme to be secure, it has to be very hard to compute p and q from N (to *factorise* N).

2. Proof of correctness

We want to show that if the conditions stated above on N, e and d are satisfied, then $\widetilde{m}^d = m \bmod N$. First note that it suffices to show that $m^{ed} = m \bmod N$. It then suffices to show that both $m^{ed} = m \bmod p$ and $m^{ed} = m \bmod q$; this is a consequence of the *Chinese reminder theorem*, and here is the relevant part of its proof from which that claim follows. Set $x = m^{ed}$, and for a contradiction assume that $x = m \bmod p$ and $x = m \bmod q$, but $x \neq m \bmod pq$. Let natural number $y < pq$ be such that $x = y \bmod pq$. Then $x = y \bmod p$ and $x = y \bmod q$, hence $m = y \bmod p$ and $m = y \bmod q$. But then $m - y$ is divisible by both p and q, so $m - y$ is also divisible by pq which since $|m - y| < N$, implies that $m = y$, yielding the desired contradiction.

Now let us prove that $m^{ed} = m \bmod p$ (the proof that $m^{ed} = m \bmod q$ is similar). If $m = 0 \bmod p$ then $m^{ed} = 0 \bmod p$ and we are done, so suppose $m \neq 0 \bmod p$. Then m^{ed} is equal to $m^{ed-1}m$, which is equal to $m^{c\phi(N)}m$ for some natural number c (because $ed = 1 \bmod \phi(N)$), which is equal to $(m^{p-1})^{c(q-1)}m$, which is equal to m modulo p if we can establish that m^{p-1} is equal to 1 modulo p. But that immediately follows from *Fermat's little theorem*, which states that $m^p = m \bmod p$ holds as a consequence of p being prime (indeed, as p does not divide m by hypothesis, $m^p = kp + m$ for some integer k implies that $m^{p-1} = k'p + 1$ for some integer k').

Let us remind the simplest proof of Fermat's little theorem. It uses a combinatorial argument. There are m^p sequences of length p of numbers from $\{1, \dots, m\}$. Of those, m consist of nothing but the same numbers: $(\overbrace{1, \dots, 1}^{p})$,

Date: Session 1, 2015.

..., $(\overbrace{m,\dots,m}^{p})$, so it suffices to show that we can put the remaining sequences in groups all of size p. Put two sequences in the same group if one can be obtained from the other by rotation, that is, if one is of the form $(x_1,\dots,x_p)$ then the other is $(x_i,\dots,x_p,x_1,\dots,x_{i-1})$ for some $i \in \{2,\dots,p\}$. Suppose for a contradiction that there is a group of size less than p. We can then choose a minimal $i \in \{1,\dots,p\}$ and a sequence $(x_1,\dots,x_p)$ in that group with $(x_1,\dots,x_p) = (x_i,\dots,x_p,x_1,\dots,x_{i-1})$. Since $x_1,\dots,x_p$ are not all identical, i is greater than 2. As p is prime, $(x_1,\dots,x_p)$ cannot be the concatenation of $\frac{p}{i-1}$ copies of $(x_1,\dots,x_{i-1})$, so there has to exist $j \in \{2,\dots,i-1\}$ with $(x_1,\dots,x_p)$ being the concatenation of one or more copies of $(x_1,\dots,x_{i-1})$ and at the end, $(x_1,\dots,x_{j-1})$. But then $(x_1\dots x_{j-1},x_1\dots x_{i-j})$ is equal $(x_1,\dots,x_i)$, and $(x_1,\dots,x_p)$ is equal to $(x_{i-j+1},\dots,x_p,x_1,\dots,x_{i-j})$, which contradicts the minimality of i.

3. Computation of d

Set $\phi(N) = n$. We have assumed that e is relatively prime to n. Let us verify the existence and unicity of d. Since there are exactly n natural numbers smaller than n modulo n, it suffices to verify that the n numbers $0e$, $1e$, $2e$,..., $(n-1)e$ are all distinct modulo n. For a contradiction, suppose that there exists $x, y \in \{0,\dots,n-1\}$ with $x < y$ and $xe = ye \bmod n$. Then $(y-x)e = kn$ for some integer k, which since e and n are relatively prime, has to be a multiple of e, contradicting the fact that $y - x$ belongs to $\{1,\dots,n-1\}$.

Saying that $ed = 1 \bmod n$ is equivalent to saying that there exists an integer x with $nx + ed = 1$. As $\gcd(n,e) = 1$, this is a particular case of *Bézout's identity*, namely, the statement that for all nonzero integers a and b, there exists integers x and y with $ax + by = \gcd(a,b)$. Bézout's identity can be proved as follows. Let nonzero integers a and b be given, and let c be a nonzero integer of the form $ax + by$ with least absolute value. By changing the signs of x and y, we can assume that c is positive. Of course, $c \leq \min(a,b)$. Then c divides a, as otherwise a would be of the form $ck + r$ with $0 < r < c$, hence $r = a - ck = a - (ax + by)k = a(1-x) + b(-yk)$, which contradicts the definition of c. Similarly, c divides b. Also, if c' divides both a and b, then c' divides c, completing the verification that $c = \gcd(a,b)$.

Now given nonzero natural numbers a and b, the computation of $\gcd(a,b)$ and two integers x and y with $\gcd(a,b) = ax+by$ can be achieved thanks to the *extended Euclidean algorithm*. Applied to $a = n$ and $b = e$, this yields a number y such that $ey = 1 \bmod n$, and d is the remainder of the division of y by n. Given $(a,0)$ as input, the extended Euclidean algorithm returns $(a,1,0)$, which is correct since $a = a \times 1 + 0 \times 0$ and $a = \gcd(a,0)$. Given (a,b) with $b \neq 0$ as input, the extended Euclidean algorithm applies itself to $(b, a \bmod b)$, which yields a triple of the form (c,x,y), and returns $(c, y, x - \lfloor a/b \rfloor y)$. So the extended Euclidean algorithm generalises the Euclidean algorithm, which for all natural numbers a and b with $b \neq 0$, computes $\gcd(a,b)$ as $\gcd(b, a \bmod b)$ (Note that in case $b > a$, these algorithms swap both arguments, and then the first argument is always greater than the second one in all recursive calls.) The Euclidean algorithm is correct because a divisor of b is a divisor of a number of the form $bk+r$ iff it is a divisor of r. The extended Euclidean algorithm is correct because of the following sequence of equalities:

$$xb + y(a \bmod b) = xb + y(a - \lfloor a/b \rfloor b) = ya + (x - \lfloor a/b \rfloor y)b$$

4. Remarks on the implementation

To compute m^e and $\widetilde{m}^d$, we use the following equalities for modular exponentiation:

$$x^{2n} \bmod p = (x^2 \bmod p)^n \text{ and } x^{2n+1} \bmod p = ((x^2 \bmod p)^n \times x) \mod p$$

We randomly generate two numbers b_1 and b_2 with $b_2 \geq 2b_1$. Then, using Eratosthenes' sieve, we generate all prime numbers at most equal to b_2. We take q to be the largest one. By Bertrand's postulate, a theorem which implies that for all natural numbers n greater than 1, there exists a prime number in $(n, 2n)$, we infer that $q > b_1$. We then take for p the largest prime number at most equal to b_1. But the range of possible values for b_2 makes the number N so generated too small not to be easily factored, so that part of the implementation is only for illustration purposes...

We encode the sequence of ascii codes of a textual file. Every ascii code fits into a 3-digit number, so we split this sequence into chunks by glueing together k 3-digit numbers, taking for k the largest number with $3k$ smaller than the number of digits in the decimal representation of N, which guarantees that the resulting message is smaller than N; for ascii codes smaller than 100, we use leading 0s. Unless the number of characters in the file to encode is a multiple of k, the last chunk of 3-digit numbers is incomplete and will start with one or more 000s; in the decoding phase, these leading 000s will be printed out as any other character, which is fine as printing out the nul character is printing out nothing, so there is no need to give a special treatment to the last chunk.